Unit I: Introduction: Understanding Computers

Chapter 4: Data Representation: How Computers Store Information

4.1: Goal:

In this chapter, you will learn how computers represent text, numbers, images, audio, and video using digital data. You will explore binary, hexadecimal, and common methods of storing information while discovering how simple patterns of zeros and ones make modern computing possible. Understanding these concepts provides the foundation for many other areas of computer science.

4.2: Objectives:

- Explain why computers use the binary number system.
- Define the terms bit, byte, and binary digit.
- Convert simple binary numbers to decimal and decimal numbers to binary.
- Describe how text is represented using character encoding systems such as ASCII and Unicode.
- Explain how images, audio, and video are stored digitally.
- Compare units of digital storage, including bytes, kilobytes, megabytes, gigabytes, terabytes, and petabytes.
- Differentiate between data size and data transfer speed.
- Explain how data compression reduces file size.
- Describe how errors can occur during data transmission and how computers detect and correct them.
- Explain why understanding data representation is important in today's digital world.

4.3: Think About It

Imagine sending your favorite photo to a friend. Within seconds, the image travels across the internet, arrives on another device, and looks exactly the same, even though computers only understand two electrical states: on and off. How can millions of colors, words, sounds, and videos all be represented using only zeros and ones? By the end of this chapter, you'll understand the surprisingly simple idea that makes all modern computing possible.

4.4: Opening Scenario

Maria is preparing a presentation for her biology class. She types her report in a word processor, inserts photographs taken with her smartphone, records a short narration, and adds a short video demonstrating an experiment. Before submitting the assignment, she uploads everything to her cloud storage account.

Although Maria sees text, colorful images, clear audio, and smooth video, her computer stores every part of the project in the same basic form: billions of tiny binary digits called bits.

The operating system, applications, and storage devices all work together to convert those bits into the words, pictures, sounds, and videos Maria sees on the screen. Understanding how this happens helps explain why files have different sizes, why some media requires more storage than others, and how computers can communicate with one another so accurately.

In this chapter, you'll discover how computers represent every type of digital information using only two symbols: 0 and 1.

4.5: Why This Chapter Matters

Every time you send a text message, stream a movie, save a document, or upload a photo, your computer is processing data. Although digital information appears as text, images, music, and video, computers store and process all of it using binary numbers. Understanding data representation helps you make informed decisions about file storage, internet usage, image quality, backups, and data security. It also provides the foundation for understanding many other computing topics, including networking, cybersecurity, databases, programming, and artificial intelligence.

Whether you become a healthcare professional, business manager, teacher, engineer, artist, or entrepreneur, you'll work with digital information every day. Knowing how computers represent and manage data will help you use technology more effectively and understand the systems that increasingly shape our personal and professional lives.

4.6: Binary

Introduction

Every photograph you take, every song you stream, every document you write, and every website you visit has one thing in common: inside the computer, they are all stored as numbers.
Unlike people, computers cannot understand letters, pictures, sounds, or videos directly. Instead, they represent all information using only two possible values: **0 and 1**
This system is called the **binary number system**, and it forms the foundation of all digital computing.

Why Only Two Numbers?

Computers are built from billions of tiny electronic switches called **transistors**. Each transistor has only two reliable states:

- On
- Off

These two electrical states are easy for electronic circuits to recognize, even when operating billions of times each second.
To represent these two states, computers use:

ELECTRICAL STATE	BINARY VALUE
Off	0
On	1

Because every transistor has only two possible states, binary is the natural language of computers.

Binary Is Everywhere

For example, when you type the word: Hello your computer does **not** actually store the letters H-E-L-L-O.
Instead, each letter is converted into a series of binary digits. The same thing happens for:

- photographs
- movies
- music
- web pages
- video games
- text messages

Everything eventually becomes long sequences of 0s and 1s.

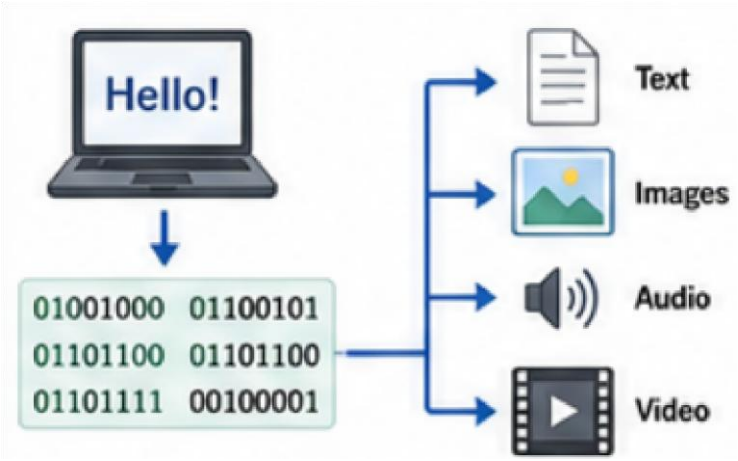


Figure: The Binary World

This illustration shows that all types of digital information, including text, images, audio, and video, are ultimately represented inside a computer as binary digits (0s and 1s). A computer converts different forms of information into binary for processing, storage, and transmission.

What Is a Bit?

A **bit** (short for **binary digit**) is the smallest unit of digital information. A bit can contain only one of two values:

0 or 1.

Think of a light switch. It can be ON or OFF.

One bit by itself cannot represent very much information.
For example:

NUMBER OF BITS	POSSIBLE VALUES
1 bit	2 values
2 bits	4 values
3 bits	8 values
4 bits	16 values

Notice that every additional bit doubles the number of possible combinations.

What Is a Byte?

Because one bit can represent only two possibilities, computers group bits together. A group of **eight bits** is called a **byte**. Example: 01000001, 1his is one byte. A single byte can represent:

- one letter
- one number
- one punctuation symbol
- many other kinds of information
- **256 different combinations**

Why Are Bytes Important? Almost every measure of computer storage begins with the byte. **For example:**

STORAGE UNIT	APPROXIMATE SIZE
Byte (B)	8 bits
Kilobyte (KB)	About 1,024 bytes
Megabyte (MB)	About 1 million bytes
Gigabyte (GB)	About 1 billion bytes
Terabyte (TB)	About 1 trillion bytes

We'll learn much more about storage units later in this chapter.

Binary Makes Digital Devices Possible

Using only two electrical states might seem limiting, but binary actually makes computers:

- reliable
- fast
- inexpensive
- accurate

Because electronic circuits only need to distinguish between ON and OFF, they are much less likely to make mistakes than if they had to recognize ten different voltage levels.
This simple idea has allowed engineers to build increasingly powerful computers for more than seventy years.

Technology in Action

Suppose you take a selfie with your smartphone.
Although you see a colorful photograph, your phone stores the image as millions of binary digits. Those bits describe:

- the color of each pixel
- the brightness of each pixel
- the file format
- the date and time
- camera settings
- location information (if enabled)

Every one of those details is ultimately represented by combinations of 0s and 1s.

Did You Know?

A modern smartphone contains **tens of billions of transistors**. Each transistor switches between two electrical states billions of times every second, allowing the device to perform everything from sending text messages to running sophisticated AI applications.

Quick Check

. Why do computers use binary instead of decimal?
2. What is a bit?
3. What is a byte?
4. How many bits are in one byte?
5. Why are transistors well suited for binary computing?

AI Activity

Ask an AI assistant such as ChatGPT, Copilot, or Gemini:
"Explain binary numbers using an everyday analogy without using any mathematical formulas." Then consider the following:

. What analogy did the AI use?
2. Did the explanation make binary easier to understand?
3. Was the explanation accurate?
4. How could you improve your prompt to make the explanation even clearer?

4.7: Base

Binary to Decimal

Decimal Number System

The decimal number system is a place system that uses a base of **10**. The right most place is the ones place. As we move from right to left, each place is **10** times the previous place.

Place:	1000	100	10	1
Digit:	1	0	4	3
Value:	1x1000	0x100	4x10	3x1

The number 1043 means 1 group of 1000, 0 groups of 100, 4 groups of 10 and 3 groups of 1. You cannot leave out the 0 in the 100s place. 1043 is not the same as 143.

In the decimal (base 10) number system, there are 10 digits: 0, 1, 2, 3, 4 5, 6, 7, 8, and 9.

Binary Number System

The binary system uses a base of 2. There are 2 digits: 0 and 1. Binary is used in computers because each bit (**b**inary dig**it**) can have just one of two values. We usually represent those two values as 1 and 0, but it can also be thought of as true or false; on or off.

The binary number system is a place system. The right most place is the ones place. As we move from right to left, each place is 2 times the previous place.

Place:	8	4	2	1
Digit:	1	0	1	1
Value:	1x8	0x4	1x2	1x1

The value of the binary number 1011 =1x8 + 0x4 + 1x2 + 1x1 = 8 +2 + 1 = 11 decimal.

For practical purposes we just add together the value of each place that has a one in it.

We read the binary number 1011 as "one zero one one." Do not say "one thousand eleven," those terms refer to the decimal system.

4.8: Convert Between Binary and Decimal

Binary to Decimal

16	8	4	2	1
1	1	0	1	1

To convert a binary number to decimal, write the place values above each digit:

Start on the right with the ones place, then multiply by 2 for each additional place: 1, 2, 4, 8,16, 32, 64, etc.

The simply add together all the places values that have a 1 in them..

For the number shown above that is 16 + 8 + 2 +1 =27.

We can write this as $11011_2=27_{10}$ In speaking we would say 11011 in base 2 is 27 in base 10.

We could also say 11011 in binary equals 27. *(If we don't specify a base it ias assumed that we mean decimal.)*

Decimal to Binary

There are two ways to convert decimal to binary.

Both start by writing the place values for binary numbers from right to left, starting with 1 and multiplying by 2 until the next place value is greater than the number we are converting.

If we want to convert the decimal number 27 we would have the values shown below. The next place value, 2x16 is 32. 32 is greaten that 27.

16	8	4	2	1

Method 1: Subtraction

For each place value from **right to left**: if the value to convert >= place value; write a 1 in the place and subtract the place value from the value to convert.

VALUE TO CONVERT	PLACE	ACTION
27	16	27>=16: write 1 in 16's place; value=27-16=11
11	8	11>=8: write 1 in 8's place; value=11-8=3
3	4	3<4: write 0 in 4's place value doesn't change
3	2	3>=2: write 1 in 2's place; value=3-2=1
1	1	1>=1: write 1 in 1's place; value=1-1=0: STOP

Result:

16	8	4	2	1
1	1	0	1	1

Method 2: Divide by 2

For each place value from **left to right**: Divide the **value** to convert by 2 (whole numbers). Write the remainder in the place.

VALUE TO CONVERT	PLACE	ACTION
27	1	27 / 2 =13 with a remainder of 1 write 1 in 1's place; value=13
13	2	13 / 2 =6 with a remainder of 1 write 1 in 2's place; value=6
6	4	6 / 2 =3 with a remainder of 0 write 1 in 4's place; value=3
3	8	3 / 2 =1 with a remainder of 1 write 1 in 8's place; value=1
1	16	1 / 2 =0 with a remainder of 1 write 1 in 16's place; value=0: STOP

Check your answer by converting the binary to decimal.

4.9: Hexadecimal

If we write a large number as a binary number, it can be quite long.
The hexadecimal, (hex) base 16, is used because every 4 binary digits can be represented by one hexadecimal digit.
In binary, base 2, there are 2 digits: 0, 1.
In decimal, base 10, there are 10 digits: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.
In hexadecimal, base 16, there are 16 digits: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.

The table below shows the corresponding values of decimal, binary, and hexadecimal:

DECIMAL	BINARY	HEXADECIMAL
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Technicians use **hexadecimal (base-16)** because it provides a shorter, easier-to-read way to represent binary numbers. Since each hexadecimal digit represents exactly four binary digits (bits), long strings of 0s and 1s can be written much more compactly. For example, the binary number **11111111** can be written as **FF** in hexadecimal. This makes it easier to read memory addresses, identify error codes, debug software, and work with computer hardware. Hexadecimal is also commonly used to represent colors in web design, where values such as **#FF0000** represent the color red.

4.10: RGB Colors

Colors on a computer screen are represented by combining three colors Red, Green, and Blue. Each dot on a screen is called a pixel (**pic**ture **el**ement).

Each pixel is a combination of 3 lights: red, green and blue.
The combination of these three colors is called the RGB color.

The value of each of the three colors can run from 0 (off) to 255 (full blast).
When all of the colors are off (0), there is no light: the color is black. When all 3 lights are on full blast (255), the color is white.
The decimal values of the colors are usually shown in hexadecimal. In hexadecimal 255 is FF.

The RGB value **FFFFFF** is white (all the pixels are on full force.)

The RGB value **000000 is BLACK** (all the pixels are off.)

The RGB value **FF0000 is RED** (the red pixel is on full force, the blue and green are off.)

The RGB value **00FF00 is GREEN** (the red pixel is off, blue is on full force, blue

is off.) The RGB value **0000FF is BLUE** (the red and green pixels are off, blue is

on full force.)

An important thing to remember is that you are mixing lights, not paint:

The RGB value **FFFF00 is YELLOW** (yellow, the red pixel and green are on full force, blue is off.)

The RGB value **FF00FF is MAGENTA** (the red and blue pixels and are on full force, green is off.)

You can experiment with RGB colors at https://zebra0.com/computer-concepts/data-representation/rgb.php

4.11: How Text Is Stored

Although we see letters, numbers, punctuation, and symbols on a screen, computers store all text as binary numbers. Each character is assigned a unique numeric value using a **character encoding** system. When you type a letter on a keyboard, the computer converts it into its corresponding numeric code, stores that code as binary, and later converts it back into a character when the text is displayed. This process happens so quickly that it appears instantaneous to the user.

One of the earliest character encoding systems was **ASCII (American Standard Code for Information Interchange)**, which assigned a unique code to 128 common characters, including uppercase and lowercase letters, numbers, punctuation marks, and control characters. As

computers became more widely used around the world, ASCII was no longer sufficient because it could not represent the thousands of characters used in different languages. Today, most computers use **Unicode**, a universal character encoding standard that supports virtually every written language, along with mathematical symbols, currency symbols, and emoji. Because of Unicode, people around the world can create, share, and display text consistently across different computers, smartphones, and websites.

ASCII Example (7-bit)

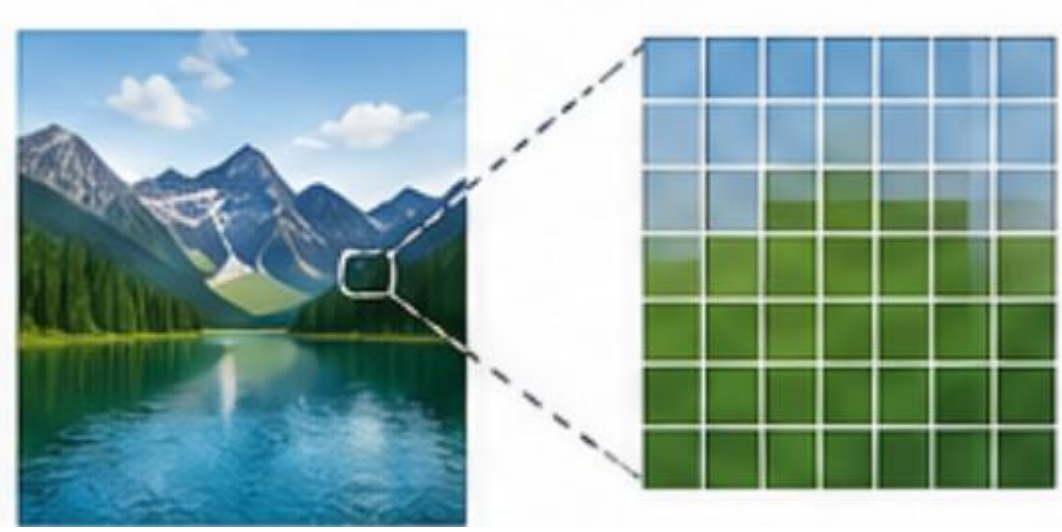
Character	Binary
A	1000001
B	1000010
a	1100001
1	0110001

Unicode Example (16-bit)

Character	Binary (hex)
A	0041
Ω	03A9
你	4F60
😀	1F642

4.12: How Images Are Stored

Digital images are made up of tiny dots called **pixels** (short for *picture elements*). Each pixel stores color information using numbers rather than paint or ink. Most color images use the **RGB (Red, Green, Blue)** color model, where different amounts of red, green, and blue light are combined to create millions of possible colors. The more pixels an image contains, the greater its **resolution**, which usually results in a sharper and more detailed picture. However, higher-resolution images also require more storage space because they contain more pixel data.

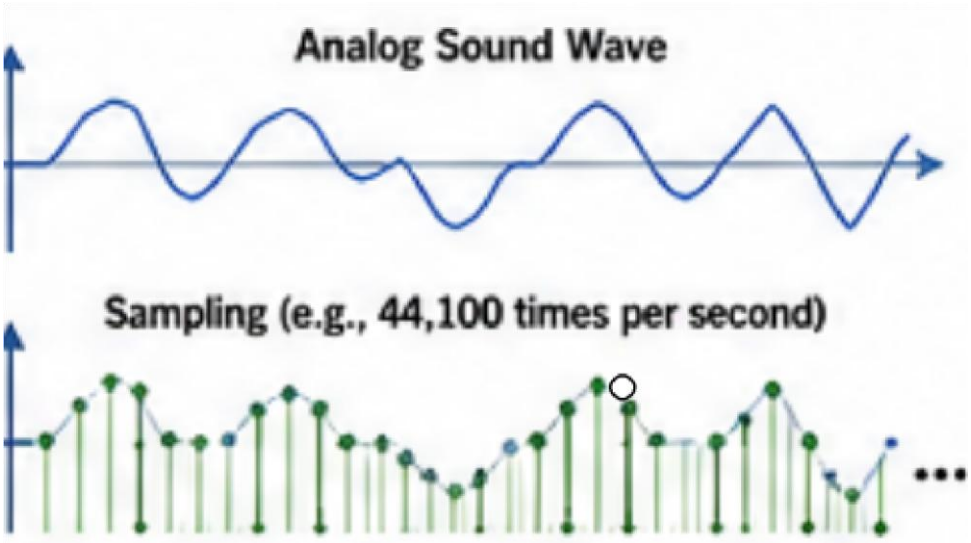


Images are saved in different **file formats**, each designed for specific purposes. **JPEG (.jpg)** files use compression to create smaller file sizes, making them ideal for photographs and sharing

images online, although some image quality may be lost. **PNG (.png)** files use lossless compression, preserving image quality and supporting transparent backgrounds, making them popular for logos, graphics, and screenshots. **GIF (.gif)** files are commonly used for simple graphics and short animations, while **SVG (.svg)** files store images as mathematical shapes instead of pixels. Because SVG images are vector graphics, they can be enlarged or reduced without losing quality, making them especially useful for illustrations, icons, and company logos.

4.13: How Sound Is Stored

Sound travels through the air as continuous waves, but computers store sound as digital data. To do this, a microphone converts sound waves into electrical signals, and an **analog-to-digital converter (ADC)** measures, or **samples**, the sound at regular intervals. The number of samples taken each second is called the **sample rate**, while the amount of information recorded for each sample is known as the **bit depth**. Higher sample rates and bit depths produce more accurate, higher-quality recordings, but they also create larger audio files. For example, music on an audio CD is typically recorded at a sample rate of **44.1 kHz** with a **16bit** depth, providing clear, high-quality sound.



How Digital Audio Is Sampled
Analog sound waves are measured (sampled) many times per second and stored as numbers. **Analog Sound Wave**
Sound in the real world is continuous and smooth, like a wave that rises and falls over time.
Sampling (Example: 44,100 times per second)
To store sound digitally, the computer measures the sound wave at regular intervals. For example, with a sample rate of **44,100 samples per second**, the computer records 44,100 measurements every second.

- Each dot represents one sample.
- The height of each sample represents the **amplitude** (loudness) of the sound at that moment.
- Each sample is converted into a binary number and stored in the computer's memory.

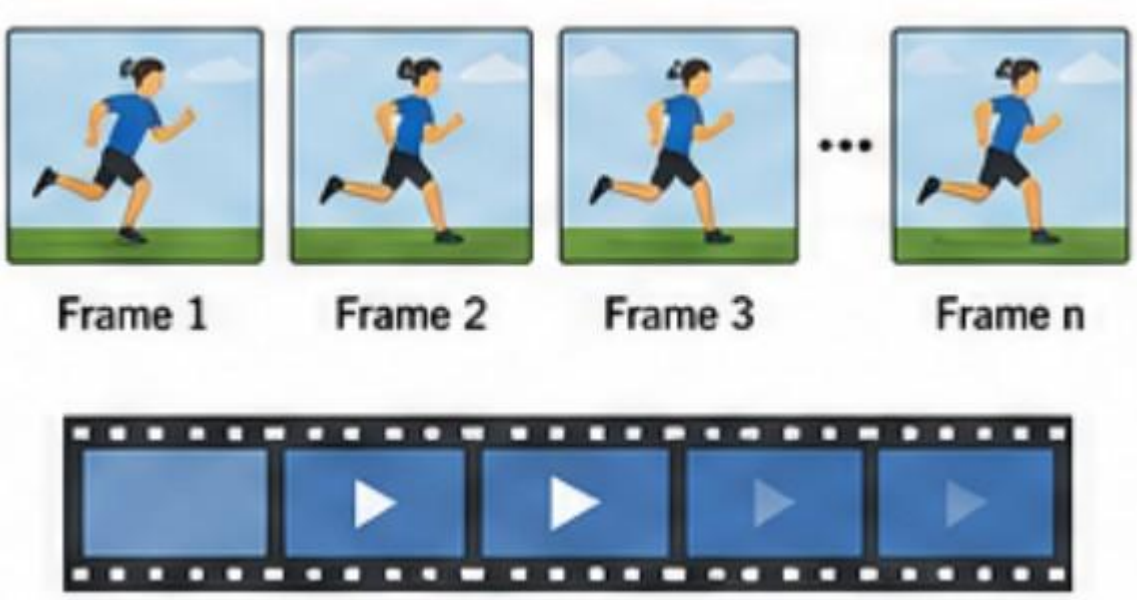
The more samples taken each second (a higher **sample rate**), the more accurately the digital recording represents the original sound wave.

Key Idea: Each sample is a number that represents the amplitude (loudness) of the sound at a specific moment in time.

Digital audio can be stored in several different file formats, each designed for a particular purpose. **WAV (.wav)** files store uncompressed audio and preserve the highest quality, making them popular for professional recording and editing. **MP3 (.mp3)** and **AAC (.aac)** files use **lossy compression** to greatly reduce file size by removing sounds that are less noticeable to the human ear, making them ideal for music streaming and portable devices. **FLAC (.flac)** files use **lossless compression**, reducing file size while preserving every detail of the original recording. Choosing the right audio format depends on the balance between sound quality, storage space, and how the audio will be used.

4.14: How Video Is Stored

Digital video is a sequence of still images, called **frames**, displayed rapidly to create the illusion of motion. Most videos are shown at **24, 30, or 60 frames per second (fps)**, meaning that dozens of individual images are displayed every second. Each frame is made up of pixels, just like a digital photograph, and usually includes synchronized digital audio.



Higher resolutions, such as **1080p (Full HD)** or **4K Ultra HD**, contain more pixels and produce sharper images, but they also require significantly more storage space.

Because uncompressed video files are extremely large, most digital videos use **compression** to reduce file size while maintaining good visual quality. Popular formats such as **MP4 (.mp4)**, **MOV (.mov)**, and **AVI (.avi)** store video using compression techniques that remove redundant information between frames. This allows movies, online videos, and video calls to be stored and transmitted much more efficiently. Streaming services such as YouTube and Netflix also compress video so it can be delivered over the internet without requiring users to download the entire file before watching.

4.15: Data Compression

Digital files such as documents, photos, music, and videos can require a large amount of storage space. **Data compression** is the process of reducing a file's size so it takes up less storage and can be transferred more quickly over the internet. Compression works by finding more efficient ways to store the data or by removing information that is less important. Smaller files save disk space, download faster, and are easier to share through email or cloud storage.

There are two main types of compression. **Lossless compression** reduces file size without losing any original information, allowing the file to be restored exactly as it was before compression. ZIP files and PNG images are common examples of lossless compression. **Lossy compression** creates even smaller files by permanently removing some data that is considered less noticeable to people. JPEG images, MP3 audio, and MP4 videos all use lossy compression to balance file size with acceptable quality. Choosing the right type of compression depends on whether preserving every detail or minimizing file size is more important.

4.16: Errors in Data Transmission

Whenever data is sent over a network or the internet, there is a small chance that some of the bits can be changed or lost because of electrical interference, weak wireless signals, damaged cables, or other transmission problems. Even changing a single bit from a **0** to a **1** (or vice versa) can cause incorrect information to be received. To help ensure accuracy, computers and networking equipment use **error detection** techniques, such as parity bits, checksums, and cyclic redundancy checks (CRC), to determine whether the received data matches what was originally sent.

If an error is detected, the receiving device may request that the data be **resent**, or it may use **error correction** techniques to recover the original information without requiring another transmission. These methods help ensure that files, emails, web pages, videos, and other digital data arrive correctly, even when they travel across long distances and many different networks. Error detection and correction are essential for reliable communication and help keep digital information accurate and complete.